МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки, специальности)

Технология программирования
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему: «Реализация криптографического алгоритма шифрования RSA»

Студент Э.Б. Асланов
(И.О. Фамилия) (личная подпись)

Руководитель к.ф.-м.н., доцент Г.А. Тырыгина
(ученая степень, звание, И.О. Фамилия)

Консультант О.А. Головач
(И.О. Фамилия)

Тольятти 2020

Аннотация

Тема бакалаврской работы: «Реализация криптографического алгоритма шифрования RSA»

Цель работы: реализация алгоритма шифрования RSA.

Объект исследования: ассиметричный алгоритм шифрования.

Предмет исследования: криптографический алгоритм RSA.

Работа посвящена исследованию и реализации алгоритма шифрования RSA. Актуальным вопросом является защита данных при передаче между пользователями Интернета.

Структура дипломной работы состоит из введения, трех глав заключение, списка литературы и приложения.

В ведении описывается актуальность проводимого исследования, формулируется цель и определяются задачи, которые необходимо решить.

В первой главе рассказывается о ассиметричной криптографии, достоинства и недостатки перед симметричными криптосистемами, о алгоритме шифрования RSA, так же его достоинства и недостатки, о всевозможных атаках.

Во второй главе описана реализация алгоритма на языке C#, средой разработкой является Visual Studio.

В третей главе проводится тестирование реализованного программного модуля.

В заключении подведены итоги выполненной работы

Бакалаврская работа представлена из 42 страницах, включает 22 иллюстраций, 1 приложение,2 таблицы, список используемой литературы содержит 30 источников ,в том числе 5 зарубежный.

Abstract

The title of the graduation work is “Implementation of a cryptographic algorithm RSA”

This graduation work is devoted to internet data protection.

The goal of the graduation work is to implement the RSA encryption algorithm.

The object of the graduation work is an asymmetric encryption algorithm.

The subject of the graduation work is cryptographic algorithm RSA.

The structure of the graduation work consists of an introduction, three chapters, a conclusion, a list of literature.

The introduction describes the relevance of the ongoing research, formulates the goal and sets tasks that need to be addressed.

The first part discusses asymmetric cryptography, the advantages and disadvantages of symmetric cryptosystems, the RSA encryption algorithm, its advantages and disadvantages, and various attacks.

The second chapter describes the implementation of the algorithm in in programming language C #. Implemented program code and graphical interface

The third chapter examines the process of testing. The program is launched, the correct operation is checked. The runtime of the program is checked.

In conclusion, the results of the work performed are summarized.

The bachelor's work is presented on 42 pages, includes 22 illustrations, 2 tables, 1 appendix, the list of used literature contains 30 sources including 5 foreign sources.

Оглавление

Введение

В настоящее время основным фактором, влияющим на политическую и экономическую, составляющие государственной безопасности, является уровень защищённости и информационной среды. Вопросы обеспечения безопасности в информационной индустрии, а также в телекоммуникационных технология актуальны и по сей день. Об этом свидетельствуют многочисленные компьютерные преступления в кредитно – финансовой сфере, так и в государственных органах.

На протяжение многих столетий человечество использовало криптографические методы защиты данных с целью обеспечить конфиденциальность при передаче информации и его хранении. Во многих источник предполагается, что криптография зародилось около 4 тыс. лет. назад. Примеры зашифрованного текста можно найти в древних манускриптах Индии и Египта.

Наука, изучающая методы зашифровки и дешифровки, называется криптология. Она подразделяется на криптографию, науку изучающая методы обеспечения целостности информации от прочтения посторонними лицами и криптоанализ, науку, изучающая способы расшифровать зашифрованную информации, не имея к этому специального ключа.

Криптография включает в себя четыре крупных раздела:

1) симметричные криптосистемы.
2) криптосистемы с открытым ключом.
3) системы электронной подписи.
4) управление ключами.

Процесс шифрования является основным видом преобразовании информации в криптографической системе. Он представляет собой преобразование открытого текста с помощью математических, логических, комбинаторных операций, в результате которых на выходе зашифрованная

информация будет представлена в виде хаотический набор букв, цифр, других символов и двоичных кодов.

Актуальность выпускной квалификационной работы заключается в том, что криптографические методы защиты информации широко применяются во многих сферах государственного, военного, коммерческого характера.

Алгоритм RSA по сей день встраиваются в частные и коммерческие продукты. Мировые компании Microsoft, Apple и Novell в своих операционных системах, так же используют RSA.Алгоритм используется в Internet и входит в протоколы такие как S/MIME, IPSEC (Internet Protocol Security) и TLS.

Объект исследования – ассиметричный алгоритм шифрования данных.

Предмет исследования – криптографический алгоритм RSA.

Цель исследования – реализация криптографического алгоритма RSA.

Для достижения цели работы необходимо решить следующие задачи:

1) Рассмотреть криптографический алгоритм.
2) Реализовать алгоритм.
3) Проверить работоспособность и корректность работы программ.
4) Выявить достоинства и недостатки.

Бакалаврская работа состоит из введения, трёх глав, заключения, списка литературы и приложений.

В первой главе затрагиваются вопросы процесса работы ассиметричных алгоритмов шифрования их достоинства и недостатки.

Во второй главе рассматривается архитектура, выбранная для решения задачи, средства реализации.

В третьей главе предоставляются результаты тестирования реализованного алгоритма шифрования на основе выявленных требований.

В заключении подводятся итоги исследования, формируются окончательные выводы и описываются результаты проделанной работы.

ГЛАВА 1 ТЕОРЕТИЧЕСКИЕ ОСНОВЫ КРИПТОГРАФИЧЕСКОГО АЛГОРИТМА RSA

1.1 Описание работы ассиметричного алгоритма шифрования

Одним из самых значимых событий в истории криптографии было создание алгоритмов ассиметричного шифрования. Алгоритмы шифрования с открытым ключом дали возможность решить две масштабные задачи, которые возникли при использовании симметричного шифрования.

Первой задачей является распределение ключей. Алгоритм симметричного шифрования предполагает о наличии общего ключа у обеих сторон, который был уже заранее передан. Американский криптограф Диффи заявил, что данное требование противоречит всей сути криптографии, о наличии конфиденциальности при коммуникациях.

Вторая задача представляет собой о создание таких механизмов, которые исключают подмену каких – либо сторон, т.е. появилась необходимость цифровой подписи. При решении задач коммерческого рода электронные сообщения и документы должны иметь эквивалент подписи, содержащейся в бумажных документах.

В 1976 г. двумя американскими криптографами У.Диффи и М.Хеллманом была предложена новая концепция криптографической системы с открытым ключом. Алгоритм представляет собой использования двух ключей. Один из ключей сделан общедоступным – открытым (public key), тем самым проблема о получении общего секретного ключа для связи отпадает [1].

Процедура шифрования использует открытый ключ. Данный ключ не является секретным и может находится в свободном доступе для всех пользователей системы, которые зашифровывают данные. Провести расшифровку данных используя открытый ключ невозможно. Для этого

используется второй ключ – закрытый ключ (private key). Пример работы ассиметричного шифрования представлен на рисунке 1.

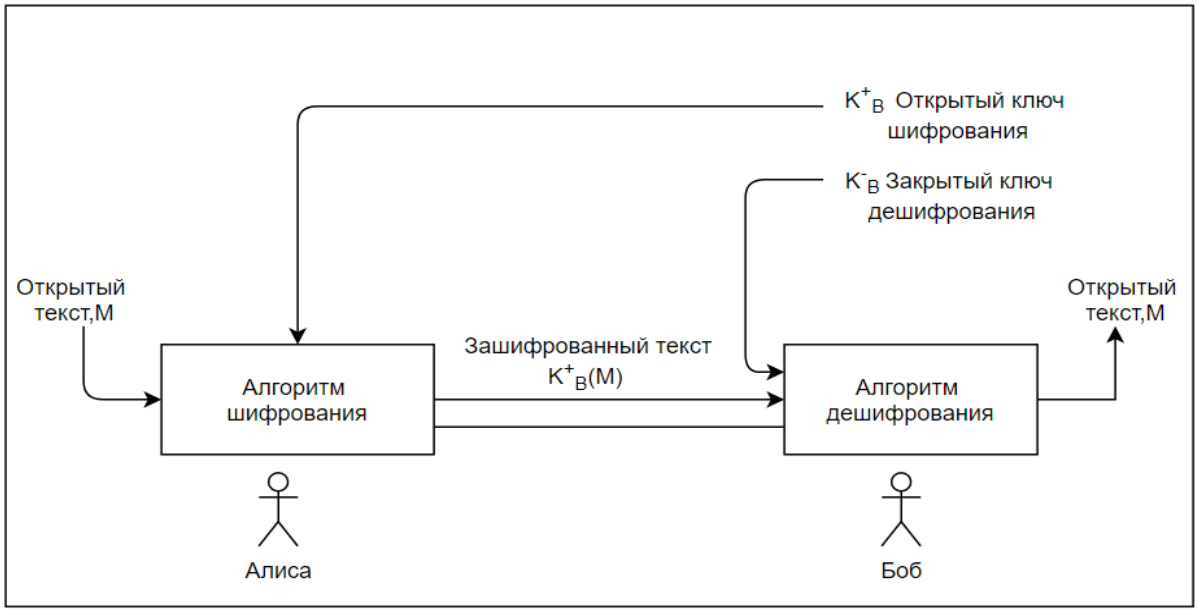

Рисунок 1 – Криптографический алгоритм с открытым ключом.

Задача строиться следующим образом. Существуют два собеседника Алиса и Боб, Алиса хочет передать Бобу некоторую секретную информацию. Далее происходит процедура генерации ключевой пары. Боб выбирает алгоритм и пару открытого ключа шифрования, и закрытый ключ дешифрования, после чего передает открытый ключ Алисе по общедоступному каналу. С помощью открытого ключа Алиса шифрует текст, который хочет передать Бобу, тем самым получая шифротекст. После чего, Боб дешифрует информацию с помощью закрытого ключа.

Преимущества асимметричных шифров перед симметричными:

– алгоритм не требует заранее переданного секретного ключа по закрытому каналу;

– лишь одной из сторон доступен ключ дешифрования, который необходимо держать в секрете;

– в сетях большого масштаба количество ключей ассиметричной криптосистеме значительно меньше, чем в симметричной.

Недостатки асимметричных шифров перед симметричными:

– внесение в алгоритм корректировок;

– длинные ключи;

– процедура шифрования – дешифрования с использование пары ключей уступает по времени симметричному алгоритму при шифровании – дешифровании того же текста;

– наличие больших вычислительных ресурсов.

Сравнительный анализ наиболее популярного симметричного алгоритма DES с асимметричным алгоритмом RSA представлена в таблице 1[3,4].

Таблица 1 – Сравнительная характеристика алгоритмов шифрования.

Алгоритм	DES	RSA
Длинна ключа	56 бит	300-1024 бит
Скорость шифрования	10 – 200 Кбайт/с	300 – 500 бит/с
Криптостйокость операций	10^{17}	10^{23}
Реализация	Программная и аппаратная	Программная и аппаратная

Исходя из таблицы можно сделать вывод, что длина ключа RSA значительно превосходит длину DES, а значит и увеличивается число комбинация подбора. Алгоритм RSA уступает в скорости шифрования, но имеет большую криптостойкость.

1.2 Система Диффи – Хеллмана

Одной из главных проблем криптографии XX века, была распределение ключей по открытым каналам. В 1976 г. появилась первая публикация статьи о криптографии с открытым ключом, авторами которого являлись У.Диффи и М.Хеллман. Дифии и Хеллман предложили алгоритм,

благодаря которому две и более стороны могли обмениваться секретным ключом по незащищенному каналу связи. Тем самым произвела большой прорыв в истории криптографии, решив задачу – распределения ключей [2].

В алгоритме Диффи – Хелламана в качестве односторонней функции используется возведение в степень по модулю простого числа:

$$y = a^x mod p, \quad (1)$$

Для организации процедуры обмена ключами между сторонами A и B, первым делом нужно выбрать два целых числа p и g, условиями которых будут таковы: p – простое, большое число, а g таково, что $1 < g < p - 1$, и все числа из множества $\{1,2, \dots, p-1\}$ могут быть представлены как различные степени g по модулю p.

Предложив, что числа p и g найдены, стороны A и B могут обмениваться ими по общедоступному каналу, так как держать их в секрете не обязательно. Далее каждая сторона выбирает случайно целое число в качестве своего секретного:

1. Сторона A вычисляет значение X и передает его стороне B:

$$X = g^x mod p \quad (2)$$

2. Сторона B вычисляет значение Y и передает его стороне A:

$$Y = g^y mod p \quad (3)$$

Значения X и Y могут передаваться по открытому каналу, поскольку получение на их основе значений секретных ключей x и y практически невозможно.

3. Сторона A вычисляет значение k:

$$k = Y^x mod p \quad (4)$$

4. Сторона A вычисляет значение k:

$$k = Y^x mod p \quad (5)$$

Только сами стороны могут вычислить значения k т.к. в вычислениях процессах используются секретные A и B. $k = g^{xy} mod p$,отсюда следует, что

k – это общий секретный ключ, который не передаётся по общедоступному каналу связи и был вычисляется сторонами независимо.

Алгоритм Диффи – Хеллмана представлен ниже на рисунке 2.

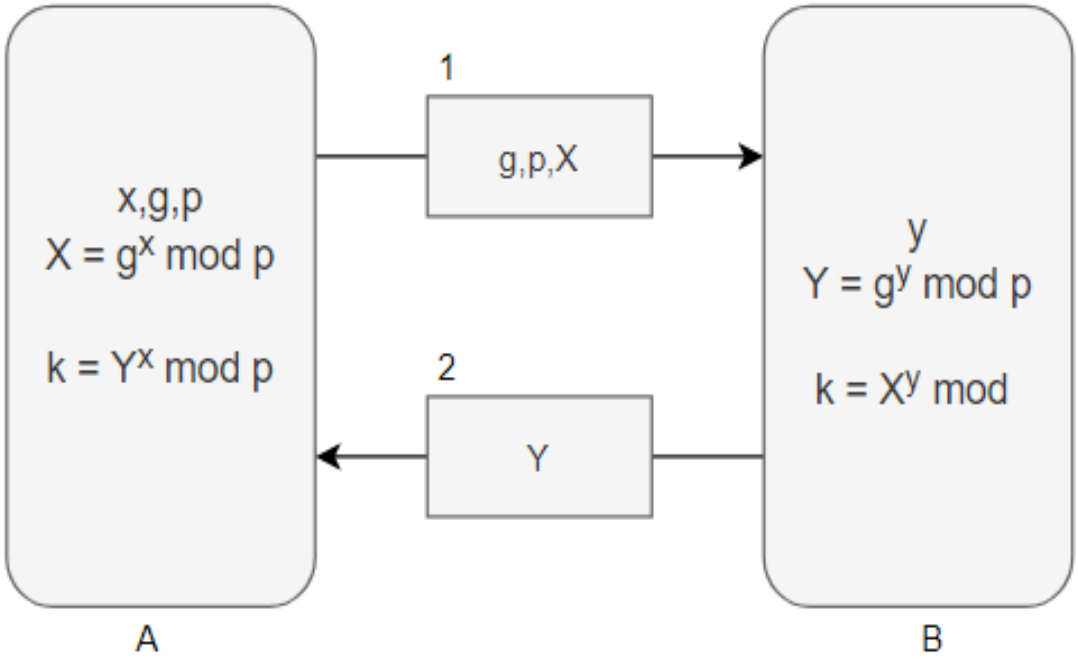

Рисунок 2 – Алгоритм Диффи–Хеллмана.

При работе алгоритма каждая сторона:

1. Производит процесс генерации случайного, натурального числа x, который является закрытым ключом.
2. Задаёт параметры для значений p и g, где p – простое, большое число,

а g первообразным корнем по модулю p.

3. Вычисляет открытый ключ X.
4. Производят обмен открытыми ключами между собой.
5. Вычисляют общий секретный ключ k

1.3 Алгоритм RSA

По сей день алгоритм RSA является одной из самых распространённых систем ассиметричного криптосистемы. Данный алгоритм был разработан в

1977г. специалистами по криптографии Р.Ривестом, А.Шамиром. и Л.Адлеманом и был назван именами своих разработчиков.

Решить проблему обменов закрытыми сообщениями позволил алгоритм Шамира, когда обе стороны способны использовать открытые каналы связи. Недостатком данного шифра является, то что сообщения пересылаются три раза между сторонами.

В свое время шифр Эль – Гамаля решает проблему обмена закрытыми сообщениями за одну пересылку, но объём передаваемого шифртоектса в два раза превышает объём сообщения.

Алгоритм RSA лишен таких изъянов. Криптографическая стойкость заключается в вычислительной сложности задачи факторизации больших чисел [4].

Предположим система состоит из трех абонентов A, B, C. Каждый абонент генерирует свою пару из личного и открытого ключа. Для этого абонент задает случайным образом два больших простых числа p и q и находит их произведение:

$$N = (p * q), \tag{6}$$

Далее абонент генерирует случайное числа e,значение которого должно быть взаимно простое со значением функции Эйлера $\varphi(N)$ от числа N:

$$\varphi(N) = (p - 1) * (q - 1), \tag{7}$$

После вычисляет значение d:

$$d = e^{-1} mod \varphi(N), \tag{8}$$

Так как числа e и $\varphi(N)$ взаимно просты, то такое число d существует и оно единственно .Алгоритм Евклида позволит найти значение числа d.

Пара (N, e) объявляется открытым ключом абонента и публикуется открыто. Значение d является личным ключом абонента и держится им в секрете. Для дешифровки сообщения абоненту достаточно знать секретный ключ. Т.к. числа $p, q, \varphi(N)$ больше не используются в дальнейших вычислительных операциях, они боле не нужны.

На основе данных вычисление, сформирована справочник открытых ключей абонентов наподобие телефонного справочника, представленный на рисунке 3.

Абонент	Открытый ключ
A	(N_A, e_A)
B	(N_B, e_B)
C	(N_C, e_C)

Рисунок 3 – Справочник открытых ключей абонентов системы RSA

Предположим абонент А хочет передать засекреченное сообщение X абоненту В.Как и во всех криптосистемах с открытым ключом сообщение М удовлетворяет неравенству $M < N_B$.

Для зашифровки сообщения абонента А выбирает из справочника пару ключей абонента В.После чего вычисляет значение С и передает его абоненту В:

$$C = M^e mod N, \quad (9)$$

Чтобы дешифровать сообщение, абонент В вычисляет:

$$F = C^d mod N, \quad (10)$$

После вычисление F на выходе получается исходное сообщение.

1.4 Цифровая подпись

Электронная цифровая подпись эквивалента подписи, которую ставят для подтверждения личности в бумажном виде, которая служит для защиты электронного документа от подделки. Цифровая подпись на основе алгоритма RSA – это практически и есть шифр алгоритма RSA, только в место сообщения шифруется дайджест и шифруется он с помощью закрытого ключа [13].

Работа алгоритма ЭЦП на основе RSA заключается в следующем:

– абонента А хэширует свой документ с помощью однонаправленной

хэш – функции;

– далее, абонент А проводит процедуру шифрования вычисленного хэш – значения;

– зашифрованное значение передается абоненту В;

– абонент В с помощью открытого ключа абонента А расшифровывает хэш – значение документа. Если после расшифровки оба значение совпали, то подпись верна.

Работа алгоритма ЭЦП на основе RSA показана на рисунке 4.

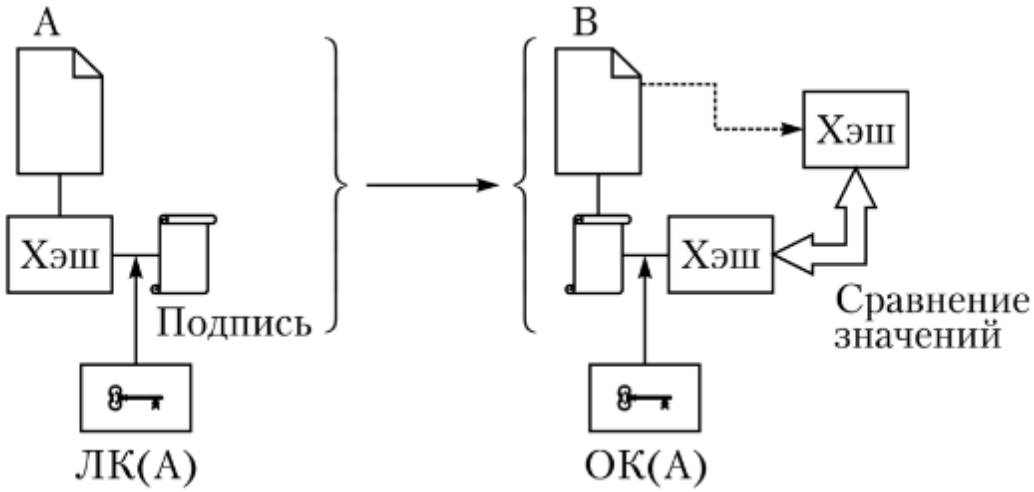

Рисунок 4 – Цифровая подпись сообщения на основе RSA

Формальное описание системы электронной цифровой подписи на основе алгоритма RSA.Предположим, некий абонент А желает отправить подписанное сообщение М абоненту В. Как и в алгоритме RSA, абонент А задает два случайных больших числа p и q,после вычисляет их произведение и значение $\varphi(N)$,далее задаёт значение для числа e и вычисляет значение d.Абонент А передает свой открытый ключа ,чтобы любой смог проверить его подпись:

1. Вычисляется хеш—значение h, которое должно удовлетворять неравенству $0 < h < n$;

2. Абонент А подписывает свое сообщение М, вычисляя значение s:

$$s = h^d mod N, \quad (11)$$

3. Получившаяся подпись добавляется к сообщению M, образуя пару (M, s), которая свою очередь передается абоненту B;

4. Любой абонент знающий открытый ключ абонента A, сможет проверить подпись на подлинность.

Электронная цифровая подпись на основе RSA является детерминированной, т.к. если сообщение подписать два раза, то на выходе будут получены одинаковые подписи [24,25].

1.5 Алгоритм Евклида

Алгоритм Евклида играет большую роль в реализации алгоритма RSA.Благодаря оценки его производительности, позволяет оценить эффективность криптографического алгоритма. Для нахождение секретного ключа, необходимо вычислить значение d ,которые должно быть взаимно простым с $\varphi(N)$,то бишь не иметь общих делителей. А так же значение d должно быть меньше $\varphi(N)$.Функцией алгоритма является вычисление наибольшего общего делителя и выполняется по рекуррентной формуле:

$$\text{НОД}(\mathrm{a}, b) = \text{НОД}(b, a \bmod b), \qquad (12)$$

Свойство алгоритма, пусть $\mathrm{c} = \text{НОД}(\mathrm{a}, b)$, а $d = \text{НОД}(b, r)$. Тогда число c делится на a и b, отсюда следует, что и c делится на r. Поэтому, $\mathrm{c} \leq \mathrm{d}$. Аналогично рассуждая, получим, что: d делится на b и r, поэтому d делится на a. И поэтом, $\mathrm{d} \leq \mathrm{c}$. В итоге получаем, что c = d. Если $\text{НОД}(\mathrm{a}, b)$=1, то числа считаются взаимно простыми. Ход работы алгоритма, для нахождения наибольшего общего делителя:

1. Первым действием выбрать два натуральных числа a и b.
2. Если b = 0, то нужно остановиться, результатом будет значение a, в противном случае перейти к следующему шагу.
3. Произвести замену значение a на значение b, а значение b остатком от деления a на b и перейти к шагу 2 [12].

В результате описания данного подраздела, был разобран алгоритм Евклида, который позволит вычислить значения секретного ключа взаимно простым с функцией Эйлера.

1. 6 Выбор параметров RSA

Криптостойкость RSA напрямую зависит от выбора параметров чисел. Алгоритм работает на основе двух пары простых больших чисел p и q.Чтобы обеспечить защиту от факторизации на числа p и q накладывают следующие требования:

- значение чисел не должны содержаться в списках известных больших простых чисел;
- исключить разложение чисел $p-1$, $p+1$, $q-1$, $q+1$ на произведение малых простых множителей;
- значение чисел не должны быть похожими, иначе злоумышленник может воспользоваться для факторизации N метод Ферма;
- числа $p_1 = \frac{p-1}{2}$, $p_1 = \frac{p+1}{2}$, $q_1 = \frac{q-1}{2}$, $q_1 = \frac{q-1}{2}$ следует выбрать простыми.

Для того чтобы получить N равную 1024 бит, нужно подобрать такие числа p и q длинна которой равны 512 бит. Например, 612 и 412 бит [15].

Далее рассмотрим выбор параметров для шифрования и расшифрования. Если несколько пользователей будут пользоваться одинаковым открытым ключом e и сообщение будет пересылаться одним и тем же пользователем одного и того же сообщения ,то они подвергаются атаке на основе китайской теореме об остатках.

Выбор маленьких значений для d ,так может подвергнуться к атаке М.Винера. Существует эффективный способ вычисление d в случае ,если $d < \frac{1}{3}\sqrt[4]{N}$.Атака Винера была обобщена на случай $d < N^{0,292}$.Исходя из этого при выборе d длинна ,значение должно иметь длину порядка 300 бит и более [16].

Выбор e так же ограничен следующими требованиями:

– выбрать e при условии $1 < e < \varphi(N)$;

– значение e должно быть взаимно простым со значением $\varphi(N)$.

Таким образом, в данном подразделе были разобраны основные условия выборов параметров для чисел p и q.А так же параметры для открытого и закрытого ключа [9].

1.7 Пример работы алгоритма RSA

Предположим необходимо зашифровать слово «SKY». Для этого каждую букву слова, нужно пронумеровать порядковым номером в английском алфавите. S –19, K – 11, Y – 25. Далее следуем алгоритму:

1. Для начала случайным образом генерируем два простых числа (для простоты решение, были выбраны небольшие числа): $p = 7, g = 13$.
2. Находим их произведение, N:

$$N = p * q = 7 * 13 = 91$$

3. Находим функцию Эйлера от модуля числа N:

$$\varphi(N) = (p - 1) * (q - 1) = (7 - 1) * (13 - 1) = 72$$

Случайным образом выбираем: $e = 5$

4. Вычисляем значение d:

$$d * e = 1\ mod(N) = d * 5 = 1\ mod(72), d = 29$$

5. Открытым ключом (5,91) шифруем каждую букву исходного сообщения.

$$a1 = 19^5 mod\ 91 = 80$$

$$a1 = 11^5 mod\ 91 = 72$$

$$a1 = 25^5 mod\ 91 = 51$$

6. Закрытым ключом дешифруем (29,91) каждую букву исходного сообщения.

$$b1 = 80^{29} mod\ 91 = 19$$

$$b1 = 72^{29} mod\ 91 = 11$$

$$b1 = 51^{29} mod\ 91 = 25$$

На рисунке 5 наглядно показан ход работы алгоритма.

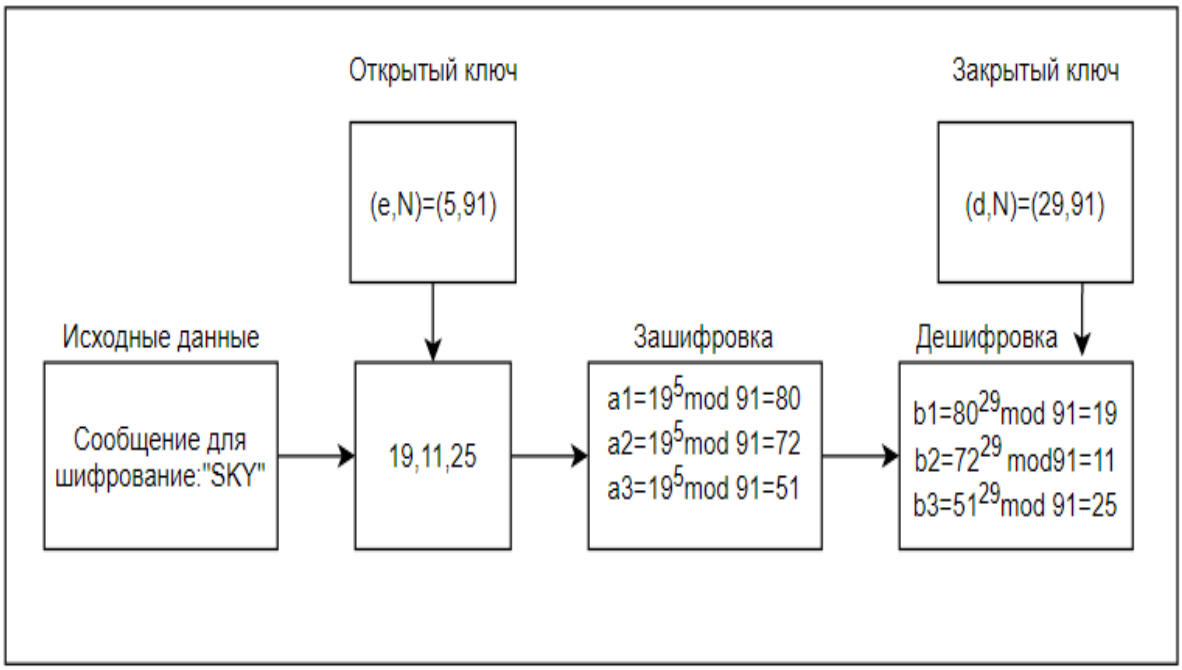

Рисунок 5 – Пример шифрования RSA

В итоге получаем зашифрованное сообщение. Имея открытый ключ, любой абонент А может отправлять зашифрованные сообщение абоненту В, но никто кроме абонента В не сможет расшифровать и прочить данные сообщения, т.к. для вычисление необходим секретный ключ d абонента.

Таким образом в данном подразделе с помощью примера было наглядно показано, как работает алгоритм шифрования RSA.

1.8 Криптоанализ RSA

Криптоанализ – это наука, которая позволяет оценивать сильные и слабые стороны методов шифрования и находить способы получение секретной информации не имея доступа.

Алгоритм RSA изобрели в 70 – х годах и используется по сей день для скрытия и защиты информации. Данная система является очень надежной

если правильно подобрать и вычислить компоненты при реализации: необходимо правильно выбрать открытую экспоненту, вычислить секретный ключ, быть уверенным, что в реализации отсутствуют недостатки, связанные с атаками по таймингу, с атаками по сторонними каналам, по кэшам и т.д. [17,20].

Отличительной чертой криптосистемы RSA от шифрования на открытых ключах является высокая производительность. Для того чтобы возвести число в степень размером 4000 бит потребуется немало производительности т.к. данная операция очень сложная. Такие криптосистемы как DES и AES используют намного меньше операций.

Чтобы повысить производительность крипосистемы был предложен другой метод. Если необходимо зашифровать сообщение, то для начала с помощью криптографически стойкого генератора случайных чисел генерируется новый ключ того размера, который необходим для шифрования блочным шифром. После чего ключ сессии возводят в секретный множитель модулю n и передается отправителю.С помощью данного ключа шифруют информацию текущей сессии, сам же ключ зашифрован алгоритмом с открытым ключом.

Данный метод подходит для передачи сообщение по открытым каналам, т.к. для начала передается зашифрованный сессионный ключ по открытому каналу, после отправитель осуществляет дешифровку данного ключа, и злоумышленник ничего не узнает. Процесс шифрования данных между собеседниками происходит эти ключом и происходит он один раз в начале сессии. Данная процедура происходит быстро т.к. объём данных небольшой.

Существует несколько способов сломать алгоритм RSA.Наиболее эффективный способ – нахождение закрытого ключа. Если злоумышленнику получиться отыскать закрытый ключ, то это даст ему доступ к прочтению зашифрованное информации. Произойдёт это лишь в том случае, если разложить n на простые множители p и q. После чего, имея данные

компоненты вычислить закрытый ключ d.Надежность алгоритма RSA зависит от разложения n [18].

Вычисление корня степени e по $mod\, n$ еще один способ сломать алгоритм RSA.Вычислив данный корень , мы получим исходное сообщение m.Данный метод не даст нам закрытый ключ d.Тем, не менее вычисление корней ,представляет сложную задачу.

1.8 Атаки на RSA

Атака путём факторизации. Факторизацией является разложение натурального числа на его произведение простых множителей. Факторизация считается сложной задачей. На данный момент времени не известно существует ли не квантовый алгоритм, который способен провести факторизации целых чисел.

Безопасность криптосистемы RSA тесно связана с задачей факторизации. Злоумышленник, который успешно сможет произвести процедуру факторизации модуля N ,получит значения для вычислений секретного ключа. После чего сможет восстановить открытый текст M, но и сможет расшифровать всю зашифрованную информацию.

Атака на основе Китайской теоремы об остатках. Китайская теорема об остатках в криптосистеме RSA, служит для повышения шифрования. Если число e задать небольшим, то скорость шифрования информации можно ускорить. Данное ускорение может привести к негативным последствиям. Но данный случай работает лишь тогда ,когда несколько абонентов для шифрования использует один и тот же закрытый ключ e.

Предположим, несколько абонентов имеют попарно взаимно простые модули N_1, N_2, N_3 и общий секретный ключ $e = 3$.Если всем пользователям было послано одно и тоже сообщение x ,то злоумышленник способен определить его. Злоумышленник перехватит все три сообщения, $C_i = M^3 mod\, N_i$,где $i = 1,2,3$.После чего он вычислит решение C системы

сравнений, на интервале $0 < C < N_1 \cdot N_2 \cdot N_3$.Исходя из китайской теоремы об остатках данное решение C,имеет единственное решение т.к. $C^3 < N_1 \cdot N_2 \cdot N_3$,то $C = M^3$,а значение M можно вычислить, путём нахождение корня

$M = \sqrt[3]{C}$.

Атака на открытую экспоненту малого размера. Зачастую, чтобы увеличить скорость шифрования и расшифрования, используют малые значения для открытого и закрытого ключа. Данный способ может привести к всевозможным атакам [17].

Если пользователь выбрал e значение которого является небольшим, то большое количество открытых сообщение, которое удовлетворяет неравенству $M < \sqrt[e]{N}$,будут зашифрованы посредством простого возведения в степень $C = M^e < N$,тем самым подвергая себя к атак. Чтобы получить эти открытые сообщения ,злоумышленнику потребуется извлечь корень степени e.При выборе малого значение секретного ключа ,система подвергается атаке Винера. Методом перебора малых значений до получения искомого d.Поэтому длина секретного ключа должно быть не менее 300 бит.

Таким обозом в данном подразделе были разобраны основные атаки способы взлома криптосистемы RSA [8,20].

1.9 Достоинства и недостатки RSA

Криптосистема RSA является одной из числа надежных систем защиты информации. Миллионы пользователей пользуются технологией RSA BSAFE для шифрования данных. Так как при использовании данной технологи, зачастую используется алгоритм RSA.Но у каждой системы есть ряд своих достоинств и недостатков.

Одним из главных достоинством алгоритма RSA является, то что невозможно произвести повторную процедуру зашифровки сообщения, имея открытый ключ и зная алгоритм шифрования. На базе алгоритма RSA

работает программа шифрования PGP, которая позволяет проводить операцию шифрования и цифровой подписи [25,26].

В симметричных криптосистемах для улучшения надежности, после каждой процедуры шифрования данных, необходимо производить замену ключа на новый, в случае с ассиметричной криптосистемой пару ключей (e, d) можно не менять длительное время.

Одним из недостатков криптосистемы RSA является низкая скорость шифрования. Алгоритм RSA уступает алгоритму DES и другим алгоритмам шифрования. Системе RSA понадобится в 2 раза больше времени, чтобы произвести операцию шифрования – расшифрования с использованием пары ключей, чем шифрование – расшифрование того же самого текста симметричным алгоритмом. К тому же процедура шифрования происходит быстрее, чем расшифровка. Чтобы повысить эффективность ассиметричных алгоритмов, производят смешивание методов [28].

Для шифрования данных используют симметричный алгоритм со случайным сеансовым ключом, а для генерации ключа используют алгоритм RSA.Данный метод имеет ряд уязвимостей т.к. для генерации такого сеансового ключа, нужен криптостойкий генератор случайных чисел, который сможет противостоять всевозможным атакам симметричного криптоалгоритма [30].

Выводы по первой главе:

При работе над первой главой данной дипломной работы были рассмотрены основные теоретические аспекты криптографического алгоритма RSA, которые понадобятся в ходе работы над ВКР.

Были рассмотрены основные понятия, такие как:

- ассиметричный алгоритм шифрования;
- система Диффи – Хеллмана;
- алгоритм RSA;
- алгоритм Евклида.

Также в ходе работы над первой главой был подготовлено задание для тестирования програмы алгоритма шифрования. Были выявлены основные достоинства и недостатки алгоритма. Также были разобраны всевозможные атаки на криптосистему RSA.

ГЛАВА 2 ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ АЛГОРИТМА RSA

Для реализации алгоритма был выбран язык C#. Язык C# считается одним из популярных языков программирования, работа с которым позволит разрабатывать программы, а также графические интерфейсы к ним.

Данный язык является объектно-ориентированным. Средой разработки является Visual Studio, так как данная среда создана специально для языка C# и хорошо подходит для поставленной задачи.

В ходе реализации будет написан программный код, а также будет реализован интерфейс. В котором пользователь сможет сам задавать криптостойкость алгоритму, а также сможет расшифровывать уже имеющий зашифрованный файл.

Для реализации алгоритма была разобрана блок-схема, представленная на рисунке 6.

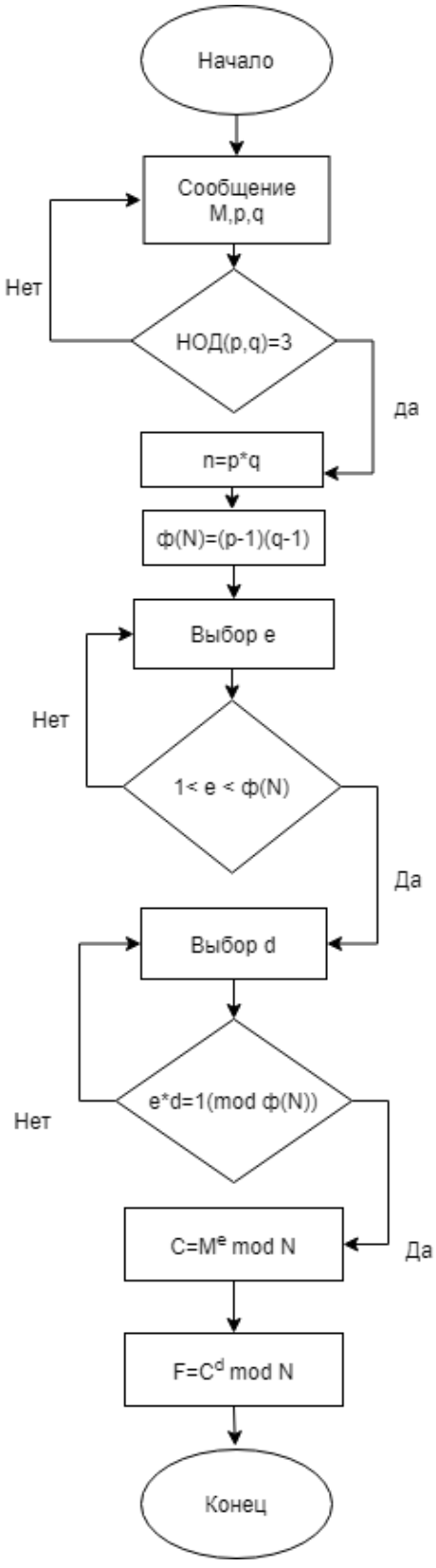

Рисунок 6 – Блок-схема работы алгоритма RSA

Входными данным алгоритма являются два простых больших числа и открытое сообщение. После чего вычисляется произведение двух простых

чисел и значение функции Эйлера. Затем подбирается e, которое должно быть взаимно простым со значение функции Эйлера. Последующим действием будет выхождение числа d, для которое действует условие. В результате на выходе будут получены пара ключей для шифрования и расшифрования. Заключительным действием будет, процедура зашифровки открытого сообщения.

Как было сказано выше, средой разработки является Visual Studio, т.к. данная среда хорошо подходит для реализации кода и соответствующего интерфейса.

Первым делом для реализации алгоритма, было подключено ряд библиотек для дальнейших действий, представленные на рисунке 7.

```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.Data;
5   using System.Diagnostics;
6   using System.Drawing;
7   using System.IO;
8   using System.Linq;
9   using System.Text;
10  using System.Threading.Tasks;
11  using System.Windows.Forms;
```

Рисунок 7 – Библиотеки

Библиотека System.Collections.Generic является одой из основных библиотек, который служит для содержания интерфейсов коллекция и классов.

Библиотека System.Windows.Forms служит для создания приложений, который позволит эффективно использовать расширенные возможности пользовательского интерфейса. Данные библиотеки являются основными для реализации интерфейса.

В программе будет использоваться следующий алфавит, представленный на рисунке 8.

```
{
    char[] alphabet = new char[] { '#', 'Й', 'Ц', 'У', 'К', 'Е', 'Н', 'Г', 'Ш', 'Щ', 'З',
                                   'Х', 'Ъ', 'Ф', 'Ы', 'В', 'А', 'П', 'Р', 'С',
                                   'О', 'Л', 'Д', 'Ж', 'Э', 'Я', 'Ч', 'С', 'М', 'И', 'Т',
                                   'Ь', 'Б', 'Ю','Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'I', 'O', 'P',
                                   'A', 'S', 'C', 'D', 'F', 'G', 'H', 'J', 'K',
                                   'L', 'Z', 'X', 'C', 'V', 'N', 'M', ',','<', '>', '/', '!',
                                   ' ','1', '2', '3', '4', '5', '6', '7',
                                   '8', '9', '0'


    };
```

Рисунок 8 – Алфавит программы.

Алфавит программы будет содержать все буквы русского алфавита и английского, так же цифры от 1 до 10. Для этого в массив alphabet записываем все буквы и символы.

Функция private void Encrypt представляет собой процедуру зашифровки. На вход подаётся файл text.txt, который записывается непосредственно через графический интерфейс в texbox1.После чего вводятся простые числа и проходит ряд последовательных действий. Ниже, на рисунке 9 приставлен фрагмент кода, отвечающий за кнопку зашифровки.

```
private void Encrypt(object sender, EventArgs e)
{
    StreamWriter ss = new StreamWriter("text.txt", false, Encoding.Default);
    ss.Write(textBox1.Text);
    ss.Close();

    if ((textBox_p.Text.Length > 0) && (textBox_q.Text.Length > 0))
    {
        long p = Convert.ToInt64(textBox_p.Text);
        long q = Convert.ToInt64(textBox_q.Text);

        if (IsTheNumberSimple(p) && IsTheNumberSimple(q))
        {
            string s = "";

            StreamReader sr = new StreamReader("text.txt", Encoding.GetEncoding(1251));

            while (!sr.EndOfStream)
            {
                s += sr.ReadLine();
            }
            sr.Close();

            long n = p * q;
            long m = (p - 1) * (q - 1);
            long d = Calculate_d(m);
            long e1 = Calculate_e(d, m);

            List<string> result = RSA_Endoce(s, e1, n);

            StreamWriter sw = new StreamWriter("EncText.txt");
            foreach (string item in result)
                sw.WriteLine(item);
            sw.Close();
```

Рисунок 9 – Фрагмент кода кнопки зашифровки

Файл text.txt является исходным открытым текстом для зашифровки. Окно textbox1 служит для ввода текста. Объект StreamWriter служит для записи данного отрытого текста. Пользователем вводится простые числа p и q.Функция SimpleValue проверяет число на простоту. Если данные значение являются простыми, то создаётся объект StreamReader который считывает записанный текст пользователем. Переменная sr хранит в себе строку, которую считал. В цикле while происходит построчное считывание, до тех пор, пока не будет достигнут конец файла.

Далее вычисляются произведения простых чисел, функцию Эйлера и пару ключей. После чего создаётся список List<string>, в котором вызывает

метод ENCR.На рисунке 10 подставлен метод, который выполняет шифрование строки. Зашифрованный текст записывается в файл Enc.txt.

```
private List<string> ENCR (string s, long e, long n)
{
    List<string> result = new List<string>();

    BigInteger var;

    for (int i = 0; i < s.Length; i++)
    {
        int index = Array.IndexOf(alphabet, s[i]);

        var = new BigInteger(index);
        var = BigInteger.Pow(var, (int)e);

        BigInteger n_ = new BigInteger((int)n);

        var %= n_;

        result.Add(var.ToString());
    }
```

Рисунок 10 – Метод, выполняющий шифрование строки

Так как при возведении числа в степень, получается большое число. Поэтому для хранение всех типов используется экземпляр класса BigInteger.Процедура шифрования происходит построчно.

Чтобы проверить числа p и q на простоту, был реализован следующий алгоритм на рисунке 11.

```
private bool SimpleValue(long n)
{
    if (n > 1)
    {
        for (int i = 2; i < n; i++)
            if (n % i == 0)

                return false;

        return true;
    }

    else
        return false;
}
```

Рисунок 11 – Фрагмент кода, проверяющий числа на простоту

Функция проверят, является ли число n простым. Если значение n >1, то цикл начинает переборку чисел от 2 до n-1. Если значение n делится без

остатка на i, то число считается не простым, если число дошло до оператора true, то оно простое. В случае, если числа не простые, вызывает окно с предупреждением, как на рисунке 12.

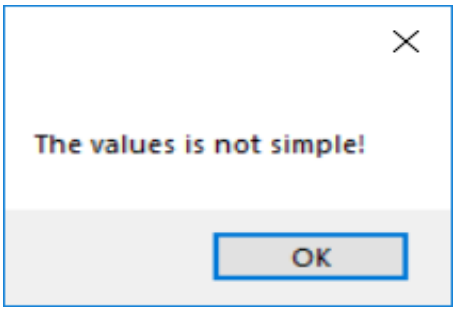

Рисунок 12 – Окно с предупреждением

Далее на рисунке 13 представлен фрагмент кода, отвечающий за работу кнопки зашифровки.

```
private void Decrypt(object sender, EventArgs e)
{
    if ((textBox_d.Text.Length > 0) && (textBox_n.Text.Length > 0))
    {
        long d = Convert.ToInt64(textBox_d.Text);
        long n = Convert.ToInt64(textBox_n.Text);

        List<string> input = new List<string>();

        StreamReader sr = new StreamReader("EncText.txt");

        while (!sr.EndOfStream)
        {
            input.Add(sr.ReadLine());
        }
        sr.Close();
        string result = DECR(input, d, n);

        StreamWriter sw = new StreamWriter("DecText.txt");
        sw.WriteLine(result);
        sw.Close();

        Process.Start("DecText.txt");

    }
}
```

Рисунок 13 – Работа кнопки расшифровки

После того, как пользователь вводит простые числа и нажимает кнопку зашифровать, в textbox_d и textbox_n будут рассчитан и выведен закрытый ключ. Далее класса StreamReader считает зашифрованный текст, ранее

записанный в EncText.txt. В цикле while происходит построчное считывание, до тех пор, пока не будет достигнут конец файла. Далее вызывает метод DECR, который производит процедуру расшифрования, представленный на рисунке 14. Полученный расшифрованный текст записывается в Dec.txt.

```
private string DECR(List<string> input, long d, long n)
{
    string result = "";

    BigInteger var;

    foreach (string item in input)
    {
        var = new BigInteger(Convert.ToDouble(item));
        var = BigInteger.Pow(var, (int)d);

        BigInteger n_ = new BigInteger((int)n);

        var %= n_;

        int index = Convert.ToInt32(var.ToString());

        result += alphabet[index].ToString();
    }

    return result;
}
```

Рисунок 14 – Метод, выполняющий расшифровку строки

Как и в методе шифрования используется экземпляр класса BigInteger. Оператор цикла foreach предназначен для перебора элементов массива. Переменная item используется в качестве интерпретатора. Переменная item приобретает значение input.

Вычисление секретного ключа представлен на рисунке 15 и 16.

```
private long valueD(long m)
{
    long d = m - 1;

    for (long i = 2; i <= m; i++)
        if ((m % i == 0) && (d % i == 0))
        {
            d--;
            i = 1;
        }
    return d;
}
```

Рисунок 15 – Вычисление значение d

Метод valueD вычисляют значение d, должно быть взаимно простым со значение m и не иметь общих делителей. Цикл будет работать пока имеют общие делители. Так же для секретного ключа необходимо вычислить значение e, представленный на рисунке 16.

```
private long valueE(long d, long m)
{
    long e = 10;

    while (true)
    {
        if ((e * d) % m == 1)
            break;
        else
            e++;
    }

    return e;
}
```

Рисунок 16 – Вычисление значение e

Метод valueE вычисляет значение e.Значение e так же должно быть взаимно простым с e.Вычисление данных компонентов даст пару ключей для шифрования исходного открытого текста.

Вывод по второй главе:

Во второй главе данной дипломной работы был реализован алгоритм шифрования RSA на языке C#.Была построена и рассмотрена блок схема алгоритм.

Реализованный программный модуль на основе данного алгоритма, выполняет следующие шаги:

- определяет введённые числа на простоту;
- на основе данных чисел генерирует открытый и закрытый ключ;
- зашифровывает открытое сообщение.
- расшифровывает зашифрованное сообщение

Основным плюсом данной реализации является, то что пользователь сам задаёт криптостойкость криптосистемы, а минусом является то что для дешифрования большого текста понадобится много времени.

ГЛАВА 3 ТЕСТИРОВАНИЕ РАЗРАБОТАННОЙ ПРОГРАММЫ

Чтобы программа эффективно функционировала без провисания и задержек времени необходимо персональный компьютер с процессором не ниже Intel Core i3, с оперативной памятью от 2 ГБ, жёсткий диск размером не меньше 500 ГБ и видеокарта от GeForce GTX 750/ Radeon R7 360 и выше.

На рисунке 17 представлена главная форма программного модуля.

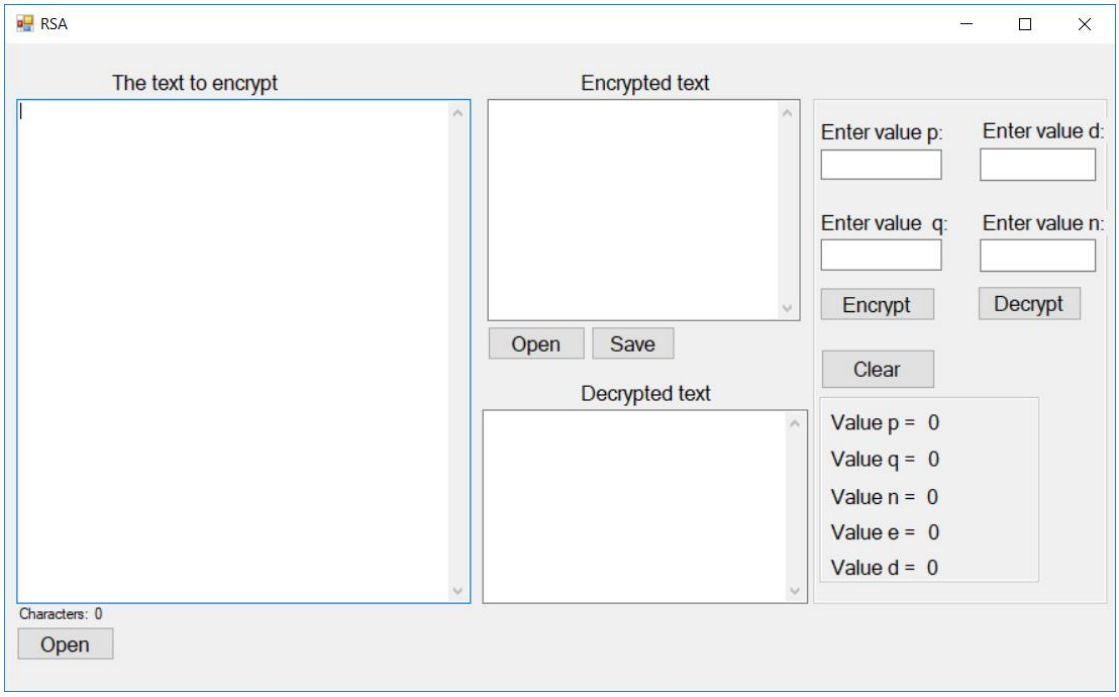

Рисунок 17 – Главная форма приложения

Главная форма состоит из кнопок «Encrypt» и «Decrypt», «Clear», «Open», «Save» окно для ввода открытого текста для шифрования, окна для ввода значения простых чисел и окна для секретного ключа и окна вывода зашифрованного и расшифрованного текста. Пользователь вводит текст или открывает уже имеющийся текст, которые хочет зашифровать, после чего вводит простые числа p и q и нажимает кнопку «Encrypt».

Если значения p и q не являются простым, то выйдет окно с предупреждением о том, что значения не являются простыми, показанное на рисунке 18.

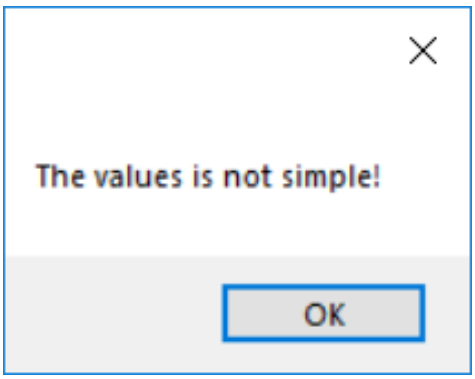

Рисунок 18 – Окно с предупреждением

Чтобы расшифровать уже имеющий зашифрованный файл пользователю нужно нажать на кнопку «Open», после чего текст зашифрованного файла будет отображен в окне «Encrypted text». Такую же процедуру можно произвести для файла, в котором находится текст для шифрования. Результат работы кнопки представлен на рисунке 19.

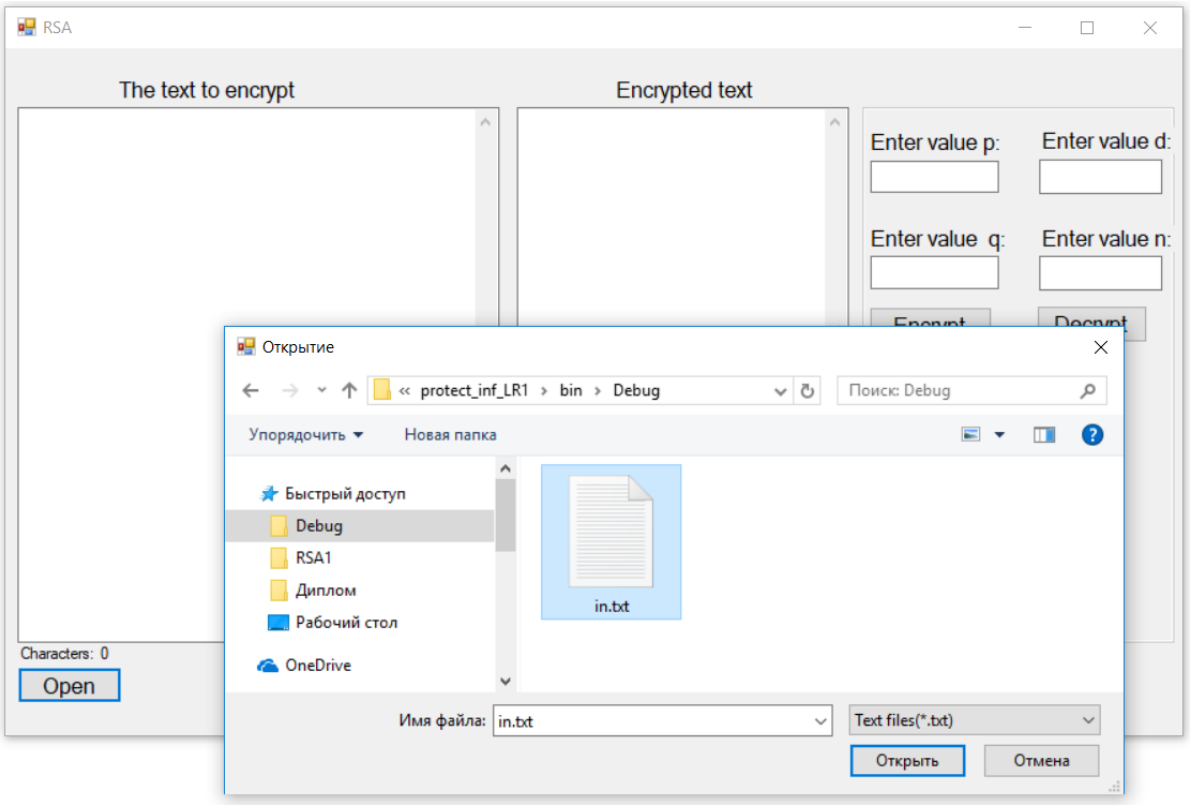

Рисунок 19 – Диалоговое окно загрузки файла

После нажатия кнопки «Encrypt» генерируются открытый и закрытый ключ. Чтобы расшифровать пользователь, должен нажать на кнопку «Decrypt».

В качестве базы для тестирования разработанного программного модуля был взят решеный пример из пункта 1.7.В таблице 2 представлены, значение простых чисел и полученные в ходе решения пары открытого и закрытого ключа.

Таблица 2 – База тестирования программного модуля.

Простые числа p,q	7,13
Произведение N	91
Открытый ключ (e, N)	(5,91)
Закрытый ключ (d, N)	(29,91)

Для шифрования было взято слово «SKY». Пример работы алгоритма приведёт на рисунке 20.

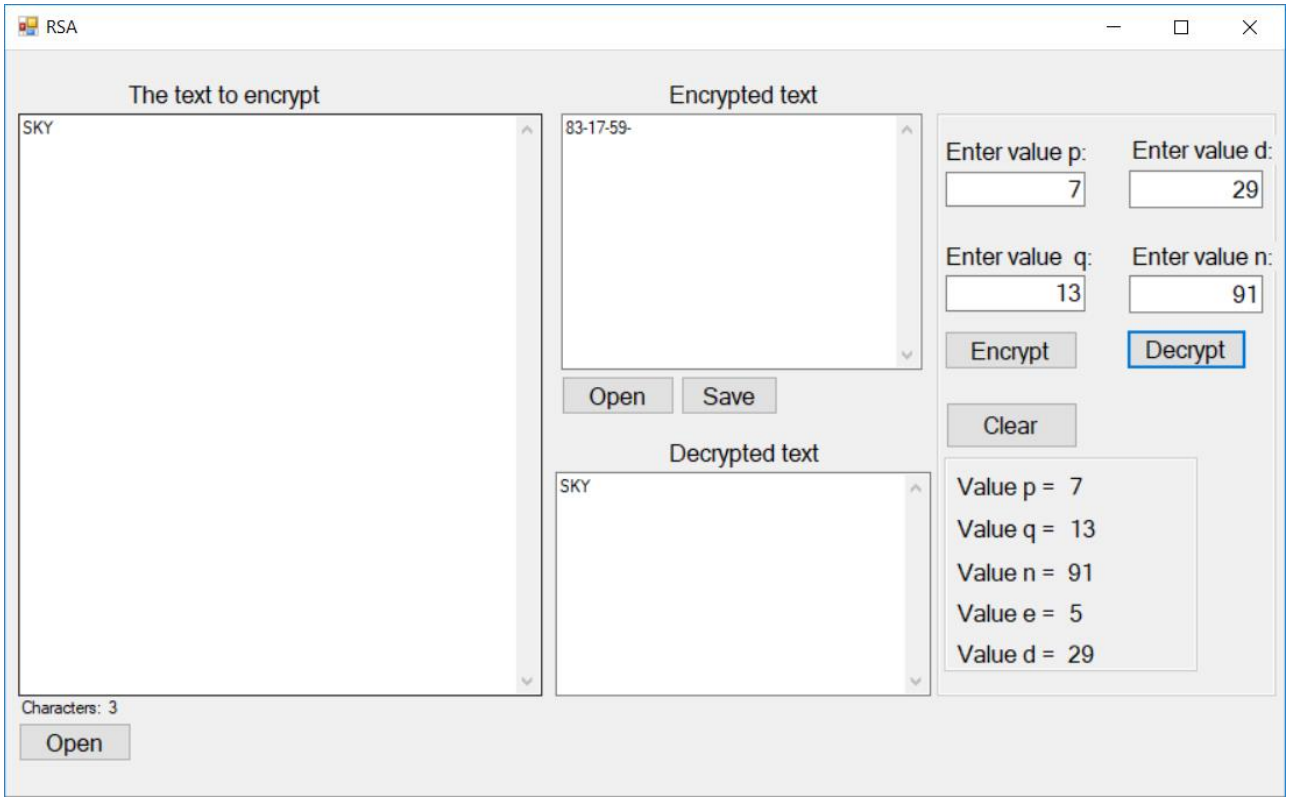

Рисунок 19 – Пример работы программы

Исходя из рисунка 19, можно сделать вывод что программный модуль работает корректно все компоненты вычислены правильно. На рисунке 20 представлен пример работы программы с текстом больше 1000 символов.

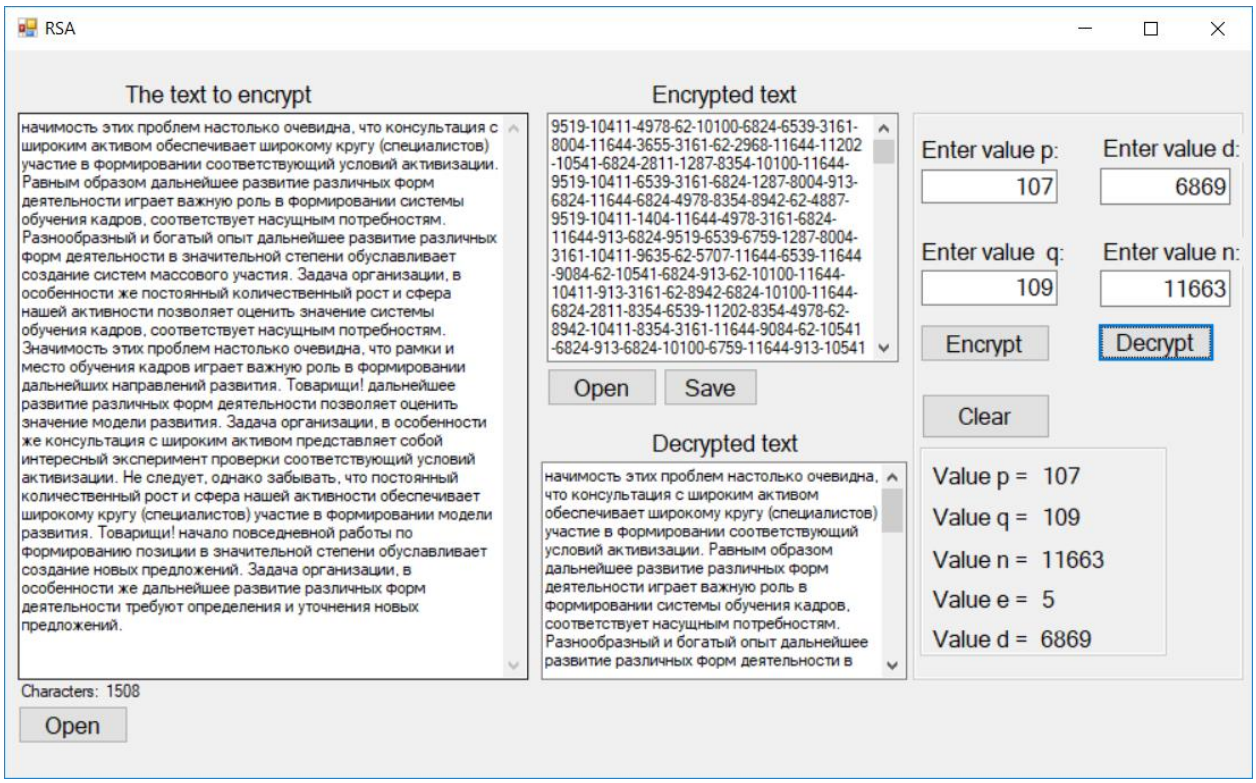

Рисунок 21 – Пример работы программы

Для тестирования времени, затраченного на шифрование открытого текста, были взяты тексты с 1000 и более знаков. В таблице 3 представлены результаты работ алгоритма с разным количеством знаков в тексте.

Таблица 3 – Время шифрования открытого текста.

Количество знаков в тексте	Время, затраченное на шифрование в мс.	Время, затраченное на расшифрование в мс.
1000	73	5654
2500	332	11236
5000	1123	24236
7500	2550	37678
10000	5136	65123

Из результатов, полученных в ходе работы программы, можно сделать вывод, что с увеличением количества знаков в тексте увеличивается и время шифрования. и расшифрования. Программе понадобится намного больше времени чтобы расшифровать.Так же влияет размер простых чисел, задаваемых пользователем.

Далее на основе данной таблицы был построен график зависимости времени, затраченного на шифрование текста, представленный на рисунке 22.

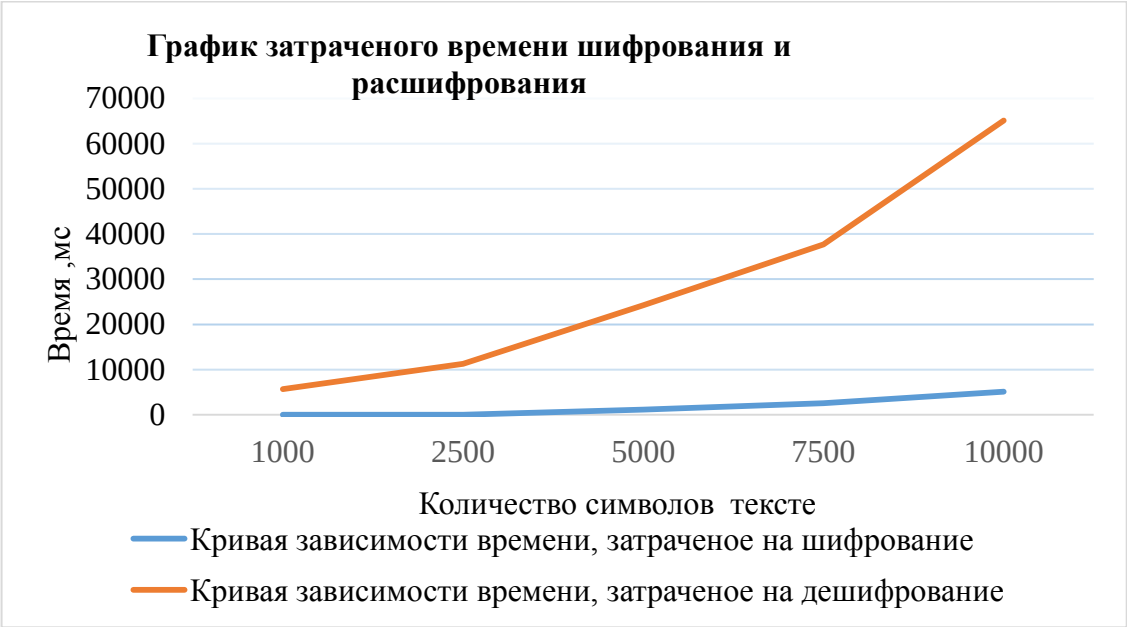

Рисунок 22 – График зависимости, затраченного времени на шифрование

В результате проведения тестирования на основе графики и таблицы, можно сделать вывод что время, затраченное на шифрование и расшифрования, зависит от количества знаков в тексте. Так же чтобы расшифровать текст понадобится значительно больше времени.

Выводы по третьей главе:

В третьей главе данной дипломной работы было проведено тестирования разработанного программного модуля. Была показана работа разработано программного модуля, выполняющая шифрования данных на основе алгоритма RSA.

Была приведена таблица и построен график, показывающий время работы алгоритма. Чем больше текст, тем больше времени будет потрачено на шифрование и расшифрование.

Заключение

В ходе выполнения бакалаврской работы был рассмотрен криптографический алгоритм RSA и был реализован программный код, который показывает работу данного алгоритма. Также был разработан графический интерфейс программного кода, чтобы пользователь без труда мог зашифровать ту или иную информацию. Были рассмотрены математические аспекты данного алгоритма, его достоинства и недостатки, а также всевозможные атаки.

Криптографическая стойкость заключается в вычислительной сложности задачи факторизации больших чисел. Атака путём факторизации, представляет наибольшую угрозу криптосистеме. Остальные атаки направлены на реализацию и недостатки алгоритма.

Данный алгоритм имеет ряд преимуществ:

– RSA является ассиметричным алгоритмом шифрования данных, а значит работает на базе передачи открытых ключей в сети. Благодаря этому пользователи способны обмениваться данными по незащищённым каналам связи.

– пользователь сам задаёт нужную ему криптостойкость, т.к. он сама задает значения чисел для p и q.

Одним из недостатков криптосистемы RSA является низкая скорость шифрования. Алгоритм RSA уступает алгоритму DES и другим алгоритмам шифрования в скорости. Системе RSA понадобится в 2 раза больше времени, чтобы зашифровать и расшифровать того же самого текста симметричным алгоритмом.

Таким образом в данной бакалаврской работе были изучены основные аспекты алгоритма RSA, выявлены основные достоинства и недостатки алгоритма, реализованный программный модуль полностью соответствует базовым механизмам алгоритма RSA.

Список используемой литературы

1. Адаменко М.В. Основы классической криптологии. Секреты шифров и кодов / Михаил Адаменко. - Москва: Высшая школа, 2014. 256с.
2. Бабаш А. В. История криптографии. Часть I / А.В. Бабаш, Г.П Шанкин. М.: Гелиос АРВ, 2002. 240 с.
3. Бабаш А.В. Криптографические методы защиты информации (для бакалавров и магистров) / А.В. Бабаш, Е.К. Баранова. - М.: КноРус, 2015. 224с.
4. Бабаш А.В. Криптографические методы защиты информации. Криптографические методы защиты информации: Учебно-методическое пособие / А.В. Бабаш. - М.: ИЦ РИОР, НИЦ Инфра-М, 2013. 413 с.
5. Баранова Е.К. Криптографические методы защиты информации. Лабораторный практикум (для бакалавров) / Е.К. Баранова, А.В. Бабаш. - М.: КноРус, 2018. 288 с.
6. Баричев С. Г. Основы современной криптографии / С.Г. Баричев, В.В. Гончаров, Р.Е. Серов. - Москва: СИНТЕГ, 2011. 176 с.
7. Вельшенбах М. Криптография на Си в действии. Учебное пособие / М. Вельшенбах. - М.: Триумф, 2014. - 462 с.
8. Герман О. Н. Теоретико-числовые методы в криптографии / О.Н. Герман, Ю.В. Нестеренко. - М.: Академия, 2012. 272 с.
9. Запечников С.В. Криптографические методы защиты информации: Учебное пособие / С.В. Запечников, О.В. Казарин, А.А. Тарасов. - Люберцы: Юрайт, 2016. 309 с.
10. Зубов А.Ю. Криптографические методы защиты информации. Совершенные шифры / А.Ю. Зубов. - М.: Гелиос АРВ, 2005. 192 с.
11. Зубов А.Н. Математика кодов аутентификации / А.Н. Зубов. - М.: Гелиос АРВ, 2014. 152 с.
12. Ишмухаметов Ш.Т. Математические основы защиты информации: учеб. пособие / Ш.Т. Ишмухаметов, Р.Г. Рубцов — Казань: Казанский федер. Унт, 2012. 138 с.

13. Коутинхо С. Введение в теорию чисел. Алгоритм RSA.Перевод с англ. С.А. Кулешова под редакцией С.К. Ландо. М. ПОСТМАРКЕТ, Москва, 2001. 328с.

14. Кузьмин Т. В. Криптографические методы защиты информации /Т.В. Кузьмин. - Москва: Огни, 2016. 192 с.

15. Литвинская О. С. Основы теории передачи информации. Учебное пособие / О.С. Литвинская, Н.И. Чернышев. - М.: КноРус, 2015. 168 с.

16. Макаров А.Теория и практика хакерских атак / Макаров А. - М.: МИК, 2015. 384 с.

17. Масленников М.Практическая криптография / М.Масленников. - М.: БХВ-Петербург, 2020. 864 с.

18. Шаньгин В. Информационная безопасность и защита информации / В.Ф. Шаньгин. - Москва: Гостехиздат, 2016. 470 с.

19. Алгоритм шифрования RSA на пальцах. [Электронный ресурс]. Режим доступа: URL: http://teh-box.ru/informationsecurity/algoritm-shifrovaniya-rsa-na-palcax.html

20. Основы криптографии: Информация. [Электронный ресурс]. Режим доступа: URL: https://www.intuit.ru/studies/courses/691/547/info

21. Gagneja K. A Survey and Analysis of Security on RSA Algorithm. / K. Gagneja, K.J. Singh. 2015. 328 с.

22. Jon C. Graff Cryptography and E–Commerce / Jon C. Graff, 2001.120 с.

23. Keqin, Feng Coding, Cryptography and Combinatorics / Keqin Feng.2015. 272 с.

24. McNeely P. Variations and Attacks on the RSA Algorithm. / P. McNeely, B. Walker. 2013. 230 с.

25. Skobic V. Hardware Modules of the RSA Algorithm. / V. Skobic, B. Dokic, Z. Ivanovic. 2014. 180 с.

Приложение А

Листинг программы реализации алгоритма RSA

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Text;
using System.Windows.Forms;
using System.Numerics;

namespace RSA
private void Encrypt (object sender, EventArgs e){
StreamWriter ss = new StreamWriter("text.txt", false, Encoding.Default);
ss.Write(textBox1.Text);
ss.Close();
if (SimpleValue(p) && SimpleValue(q)){
StreamReader sr = new StreamReader("text.txt", Encoding.GetEncoding(1251))
while (!sr.EndOfStream){
s += sr.ReadLine() }
sr.Close();
                long n = p * q;
                long m = (p - 1) * (q - 1);
                long d = valueD(m);
                long e1 = valueE(d, m);
 List<string> result = ENCR(s, e1, n);
StreamWriter sw = new StreamWriter("EncText.txt");
foreach (string item in result)
sw.WriteLine(item)
sw.Close();
                Process.Start("EncText.txt"); }
            else
                MessageBox.Show("The values is not simple!");
        }
        else
            MessageBox.Show("Entered value p и q!");}
```